(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－227403

(43)公開日　平成８年(1996)９月３日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16	３７０		Ｇ０６Ｆ	15/16	３７０Ｎ
	9/46	３４０			9/46	３４０Ｂ
	12/02	５４０			12/02	５４０
	13/00	３５５	7368－5Ｅ		13/00	３５５
// Ｇ０６Ｆ	9/40	３１０			9/40	３１０Ｃ

審査請求　未請求　請求項の数10　ＦＤ　（全 14 頁）

(21)出願番号　特願平7－207405

(22)出願日　平成７年(1995)７月24日

(31)優先権主張番号　２７９０９３

(32)優先日　1994年７月22日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国　94043　カリフォルニア州・マウンテンビュー・ガルシア　アヴェニュウ・2550

(72)発明者　パナギオティス・エス・クーギオリス
アメリカ合衆国　94040　カリフォルニア州・マウンテンビュー・デイル　アヴェニュ・ナンバー132・1200

(74)代理人　弁理士　山川　政樹

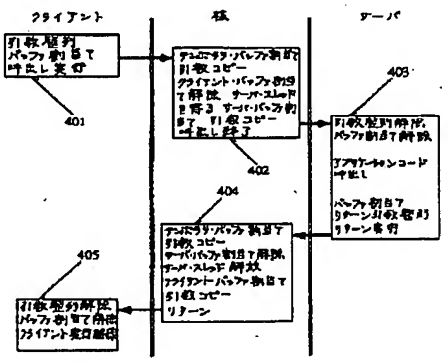
最終頁に続く

(54)【発明の名称】　スペース効率に優れたプロセス間通信方法及び装置

(57)【要約】

【課題】　スペース効率の高いプロセス間通信方法及び装置を提供する。

【構成】　第１のプロシージャが、この第１のプロシージャと第２のプロシージャによって共用される第１のメモリスペース中に第１のバッファを割当てる。第１のプロシージャは、第２のプロシージャと通信するための引数を第１のバッファ中に整列させる、第２のプロシージャのためのメッセージが現在渡されているということを指示するとともに、第１のメモリスペース中の第１のバッファに対する第１の参照を第２のプロシージャに渡す。第２のプロシージャは、第１のプロシージャによるメッセージの指示を検出した後、第１のバッファを参照し、第１のバッファ中の引数をテンポラリ・バッファにコピーする。次に、第２のプロシージャは、第１のバッファを割当て解除する。



1

【特許請求の範囲】

【請求項１】　ａ．クライアント・プロシージャが、そのクライアント・プロシージャとカーネル・プロシージャによって共用される第１のメモリスペース中に第１のバッファを割当てるステップと、

ｂ．上記クライアント・プロシージャが、リモート・プロシージャの呼出しのための引数を上記第１のバッファに整列させるステップと、

ｃ．上記クライアント・プロシージャが、上記カーネル・プロシージャを介して上記リモート・プロシージャを呼出し、上記第１メモリスペースにおける上記第１のバッファに対する第１の参照を上記カーネル・プロシージャに渡すステップと、

ｄ．上記カーネル・プロシージャが、上記クライアント・プロシージャによる上記リモート・プロシージャの上記呼出しを検出し、そのカーネル・プロシージャと上記サーバ・プロシージャによって共用される第２のメモリスペース中に第２のバッファを割当てるステップと、

ｅ．上記カーネル・プロシージャが、上記第１のバッファを参照し、上記第１のバッファ中の上記引数を上記第２のバッファにコピーするステップと、

ｆ．上記カーネル・プロシージャが、上記第１のバッファを割当て解除するステップと、

ｇ．上記カーネル・プロシージャが、上記リモート・プロシージャを呼出し、上記第２のバッファに対する第２の参照をそのリモート・プロシージャに渡すステップと、

ｈ．上記リモート・プロシージャが、そのリモート・プロシージャの上記呼出しを検出し、上記第２のバッファ中の上記引数を整列解除するステップと、

ｉ．上記リモート・プロシージャが上記第２のバッファを割当て解除し、実行するステップと、を具備したコンピュータシステムにおけるコンピュータ実装型のプロセス間通信方法。

【請求項２】　ａ．上記リモート・プロシージャの実行終了と同時に、そのリモート・プロシージャが上記第２のメモリスペース中に第３のバッファを割当てるステップと、

ｂ．上記リモート・プロシージャが、そのリモート・プロシージャからリターンするために、上記第３のバッファにリターン引数を整列させるステップと、

ｃ．上記リモート・プロシージャが、上記カーネル・プロシージャにリターンし、そのカーネル・プロシージャに上記第３のバッファに対する第３の参照を渡すステップと、

ｄ．上記カーネル・プロシージャが、上記リモート・プロシージャの上記リターンを検出し、上記第１のメモリスペースの中に第４のバッファを割当てるステップと、

ｅ．上記カーネル・プロシージャが、上記第３のバッファ中の上記リターン引数を上記第４のバッファにコピーするステップと、

ｆ．上記カーネル・プロシージャが、上記第３のバッファを割当て解除するステップと、

ｇ．上記カーネル・プロシージャが、上記クライアント・プロシージャにリターンし、上記第４のバッファに対する参照を上記クライアント・プロシージャに渡すステップと、

ｈ．上記クライアント・プロシージャが、上記クライアント・プロシージャへのリターンを検出し、上記第４のバッファ中の上記引数を整列解除するステップと、

ｉ．上記クライアント・プロシージャが、上記第４のバッファを割当て解除すし、実行を続けるステップと、をさらに具備した請求項１記載の方法。

【請求項３】　ａ．第１のプロシージャが、その第１のプロシージャと第２のプロシージャによって共用される第１のメモリスペース中に第１のバッファを割当てるステップと、

ｂ．上記第１のプロシージャが、上記第２のプロシージャとの通信ための引数を上記第１のバッファに整列させるステップと、

ｃ．上記第１のプロシージャが、上記第２のプロシージャに関するメッセージを指示し、上記第１のメモリスペース中の上記第１のバッファに対する第１の参照をその第２のプロシージャに渡すステップと、

ｄ．上記第２のプロシージャが、上記第１のプロシージャによる上記メッセージの上記指示を検出するステップと、

ｅ．上記第２のプロシージャが、上記第１のバッファを参照し、その第１のバッファ中の上記引数をテンポラリ・バッファにコピーするステップと、

ｆ．上記第２のプロシージャが、上記第１のバッファを割当て解除するステップと、を具備したコンピュータシステムにおけるコンピュータ実装型のプロセス間通信方法。

【請求項４】　ａ．上記第２のプロシージャが、上記テンポラリ・バッファ中の上記引数を処理するステップと、

ｂ．上記引数の処理の終了と同時に、上記第２のプロシージャが、上記第１のメモリスペース中に第２のバッファを割当てるステップと、

ｃ．上記第２のプロシージャが、その第２のプロシージャからリターンするために、リターン引数を上記第２のバッファに整列させるステップと、

ｄ．上記第２のプロシージャが、上記第１のプロシージャにリターンし、その第１のプロシージャに上記第２のバッファに対する第２の参照を渡すステップと、

ｅ．上記第１のプロシージャが、その第１のプロシージャへのリターンを検出し、上記第２のバッファ中の上記引数を整列解除するステップと、

ｆ．上記第１のプロシージャが、上記第２のバッファを

割当て解除し、実行を続けるステップと、をさらに具備した請求項３記載の方法。

【請求項５】　ａ．第１のプロシージャと第２のプロシージャとによって共用される第１のメモリスペース中に第１のバッファを割当てる第１の回路と、

ｂ．上記第２のプロシージャと通信するために上記第１のプロシージャによって参照された引数を上記第１のバッファに整列させる第２の回路と、

ｃ．上記第２のプロシージャに関するメッセージを指示し、上記第１のメモリスペース中の上記第１のバッファに対する第１の参照を上記第１のプロシージャからその第２のプロシージャに渡す第３の回路と、

ｄ．上記第１のプロシージャによる上記メッセージの上記指示を検出する第４の回路と、

ｅ．上記第１のバッファを参照し、上記第１のバッファ中の上記引数を上記第２のプロシージャによって使用されるテンポラリ・バッファにコピーする第５の回路と、

ｆ．上記第１のバッファを割当て解除する第６の回路と、を具備したプロセス間通信装置。

【請求項６】　ａ．上記テンポラリ・バッファ中の上記引数を処理する第７の回路と、

ｂ．上記引数の処理終了と同時に動作して、上記第１のメモリスペース中に第２のバッファを割当てる第８の回路と、

ｃ．上記第２のプロシージャからリターンするために上記第２のバッファにリターン引数を整列させる第９の回路と、

ｄ．上記第１のプロシージャにリターンし、上記第２のバッファに対する第２の参照をその第１のプロシージャに渡す第１０の回路と、

ｅ．上記第１のプロシージャへのリターンを検出し、上記第２のバッファ中の上記引数を整列解除する第１１の回路と、

ｆ．上記第２のバッファを割当て解除し、実行を続ける第１２の回路と、をさらに具備した請求項５記載の装置。

【請求項７】　ａ．第１のプロシージャが、その第１のプロシージャと第２のプロシージャによって共用される第１のメモリスペース中に第１のバッファを割当てるステップと、

ｂ．上記第１のプロシージャが、上記第２のプロシージャとの通信引数を上記第１のバッファに整列させるステップと、

ｃ．上記第１のプロシージャが、上記第２のプロシージャに関するメッセージを指示し、上記第１のメモリスペース中の上記第１のバッファに対する第１の参照をその第２のプロシージャに渡すステップと、

ｄ．上記第２のプロシージャが、上記第１のプロシージャによる上記メッセージの上記指示を検出するステップと、

ｅ．上記第２のプロシージャが、上記第１のバッファを参照し、その第１のバッファ中の上記引数をテンポラリ・バッファにコピーするステップと、

ｆ．上記第２のプロシージャが、上記第１のバッファを割当て解除するステップと、を具備したプロセス間通信方法を実装したコンピュータシステム。

【請求項８】　ａ．上記第２のプロシージャが、上記テンポラリ・バッファ中の上記引数を処理するステップと、

ｂ．上記引数の処理の終了と同時に、上記第２のプロシージャが、上記第１のメモリスペース中に第２のバッファを割当てるステップと、

ｃ．上記第２のプロシージャが、その第２のプロシージャからリターンするために、リターン引数を上記第２のバッファに整列させるステップと、

ｄ．上記第２のプロシージャが、上記第１のプロシージャにリターンし、その第１のプロシージャに上記第２のバッファに対する第２の参照を渡すステップと、

ｅ．上記第１のプロシージャが、その第１のプロシージャへのリターンを検出し、上記第２のバッファ中の上記引数を整列解除するステップと、

ｆ．上記第１のプロシージャが、上記第２のバッファを割当て解除し、実行を続けるステップと、をさらに具備した請求項７記載の方法を実装したコンピュータシステム。

【請求項９】　ａ．第１のプロシージャと第２のプロシージャによって共用された第１のメモリスペースの中に第１のバッファを割当てる第１の割当て回路と、

ｂ．上記第１のプロシージャが、上記第２のプロシージャと通信する引数を上記第１のバッファに整列させることを可能にする第１の整列回路と、

ｃ．上記第１のプロシージャからのメッセージを上記第２のプロシージャに指示し、上記第１のメモリスペース中の上記第１のバッファに対する第１の参照を上記第２のプロシージャへ渡す第１のメッセージ指示回路と、

ｄ．上記第２のプロシージャが上記第１のプロシージャによる上記メッセージの上記指示を検出することを可能にする第１のメッセージ検出回路と、

ｅ．上記第２のプロシージャが上記第１のバッファを参照し、上記第１のバッファ中の上記引数をテンポラリ・バッファにコピーすることを可能にする第１の参照回路と、

ｆ．上記第１のバッファ中の上記引数の上記テンポラリ・バッファへのコピーの終了と同時に動作して、上記第２のプロシージャが上記第１のバッファを割当て解除することを可能にする第１の割当て解除回路と、を具備したコンピュータシステム。

【請求項１０】　ａ．上記第２のプロシージャが上記テンポラリ・バッファ中の上記引数を処理することを可能にする処理回路と、

ｂ．上記引数の処理終了と同時に動作して、上記第２のプロシージャが上記第１のメモリスペースの中に第２のバッファを割当てることを可能にする第２の割当て回路と、

ｃ．上記第２のプロシージャが、その第２のプロシージャからリターンするために、上記第２のバッファ中にリターン引数を整列させることを可能にする第２の整列回路と、

ｄ．上記第２のプロシージャが上記第１のプロシージャにリターンし、上記第２のバッファに対する第２の参照をその第１のプロシージャに渡すことを可能にするリターン回路と、

ｅ．上記第１のプロシージャがその第１のプロシージャへの上記リターンを検出し、上記第２のバッファ中の上記引数を整列解除することを可能にする第２の検出回路と、

ｆ．上記第１のプロシージャが上記第２のバッファを割当て解除し、実行を続けることを可能にする第２の割当て解除回路と、をさらに具備した請求項９記載のコンピュータシステム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、コンピュータシステムに関するものである。より詳しくは、本発明は、コンピュータシステムにおけるプロセスの間通信に関する。

【０００２】

【従来の技術】プロセス間通信は、現代のコンピュータシステム設計の一つの基本的な部分である。プロセス間通信は、通常、クライアントとサーバ・プロセスとの間の通信を管理するコンピュータシステムのカーネルの呼出し技法によって容易に行える。このようなプロセス間通信に付随する問題の１つは、例えばクライアントタスクは、サーバタスクを呼び出すとき引数（パラメータ）をそのサーバタスクに渡すためにメモリの一定量を割当て、そのメモリは通常何らかの引数がサーバから戻されるまで使用状態に置かれるということである。すなわち、制御及び処理がサーバ・アプリケーションに渡された後も、サーバからリターン引数が戻されるまで、依然としてメモリがクライアント・アプリケーションで費消される。このように、サーバタスクがクライアントから制御を渡された時間の大半の間にわたって使用されることのなく一つのバッファが割当てられている。これは、各スレッドにつき１つずつというように複数のバッファが割当てられるマルチスレッド環境においては、特に問題となる。すなわち、多数のバッファが割当て状態のままに置かれ、その大部分が各スレッドの持続時間の間未使用状態に保たれ、メモリ資源が不必要に費消される。

【０００３】従来技術におけるプロセス間通信の典型的な仕組みを図１に示す。典型的には、クライアントタスク（例えば図１の１１０）は、通常、クライアント・プロセス及びカーネル・プロセスに利用可能なバッファまたは他の保護メモリスペースである符号１１１で示すようなある量のメモリスペースを割当て、そのバッファ領域に引数を整列させる。本願において、「整列」とは、クライアント・プロセスがサーバ・プロセスに渡される引数、パラメータまたは他のデータをあるメモリエリアに格納するプロセスを指す。セキュリティないしは安全保護の理由から、このメモリエリアは、通常クライアント・プロセス１１０及びオペレーティング・システム１００のカーネルのみが利用可能である。

【０００４】図１に示すサーバ・プロセス１２０に対するコールが起こると、カーネル１００がそのコールを検出し、制御はカーネルに渡される。この場合、クライアントは、通常ポインタまたは参照（reference ）をメモリエリア（バッファ）１１１に渡し、その後、カーネルはバッファ１１１に渡された引数にアクセスすることができる。次に、カーネル１００は、そのメモリエリア１１１に対する参照を受け取り、メモリエリア１１１内にある引数をカーネル１００のテンポラリ・メモリエリアにコピーする。すると、カーネルがアクセス可能な第２のメモリエリア１０１及びサーバルーチン１２０を用いて、カーネル１００からサーバ１２０へ通信することができる。テンポラリ・バッファ中の引数は、メモリエリア（バッファ）１２１にコピーされる。次に、プロセス１２０用のサーバルーチン・スレッドが生成され、カーネル１００によってこのスレッドに対する参照が行われる。サーバ１２０は、カーネル１００からのメモリエリア１２１に対する参照を受け入れ、かつ引数を整列解除する。

【０００５】サーバ・プロセス１２０がアクティブのときは、カーネル１００による呼出し後も、バッファ１１１はクライアント・プロセス１１０に割当てられた状態になっている。一部の従来技術のアプリケーションにおいては、引数を整列させるためのバッファは、長さが５キロバイトのオーダーである。クライアントは、このメモリエリアをサーバ・プロセス１２０の実行期間中、オープンでアクセス可能な状態に保つ。サーバ・プロセス１２０の実行終了と同時に、上記のクライアント／サーバ呼出しプロセスの逆のプロセスが行われ、その際、サーバは、それ自身のバッファ１２１を用いて引数をカーネル１００のメモリエリアに整列させるとともに、参照がこのカーネル１００のメモリエリアに渡される。最終的には、リターン引数は、クライアント・プロセス１１０中の元のバッファ領域１１１内に戻る。典型的な従来技術のシステムにおいては、この時になって初めて、クライアント・プロセス及びサーバ・プロセスのバッファ１１１及び１２１が両方とも割当て解除される。このように、これらのバッファは、カーネル１００及びサーバ・プロセス１２０がアクティブであり、またおそらくはアイドルである（例えばＩ／Ｏ処理待ちの間）長い時間

にわたって、割当てられたまま、アイドル状態に保たれることになる。これは、メモリ資源の無駄遣いである。さらに、マルチスレッド環境においては、クライアント・プロセス１１０及びサーバ・プロセス１２０が１１１及び１２１のようなバッファを多数割当てて、対応する呼び出された側のプロセスによって制御がリターンされるのを待つ間、それらのバッファを割当て状態に保つ場合がある。これは、メモリ資源が大量かつかつ不必要に費消される結果につながる。

【０００６】

【発明が解決しようとする課題】コンピュータシステムにおけるスレッド数が増大するにつれて、このような通信によって費消されるメモリの量が非常に重要な意味を持つようになる。従って、本発明は、プロセス間通信におけるメモリ資源の使用に付随する従来技術の欠点を解消するためになされたもので、その目的は、スペース効率の高いプロセス間通信方法及び装置を提供することにある。

【０００７】

【課題を解決するための手段】本発明は、コンピュータシステムにおけるプロセス間通信用のコンピュータで実行される方法及び装置にある。本発明においては、第１のプロシージャが、この第１のプロシージャ（例えばクライアント・プロセス）と第２のプロシージャ（例えばカーネルまたはサーバ・プロセス）によって共用される第１のメモリスペース中に第１のバッファを割当てる。次に、第１のプロシージャは、第２のプロシージャと通信するための引数を第１のバッファ中に整列させる。また、第１のプロシージャは、第２のプロシージャのためのメッセージが現在渡されているということを指示するとともに、第１のメモリスペース中の第１のバッファに対する第１の参照を第２のプロシージャに渡す。第２のプロシージャは、第１のプロシージャによるメッセージの指示を検出する。次に、第２のプロシージャは、第１のバッファを参照し、第１のバッファ中の引数をテンポラリ・バッファにコピーする。すると、第２のプロシージャは、第１のバッファを割当て解除することができる。本発明の実施の形態においては、第１のバッファは第２のプロシージャによる通信の受取りと同時に割当て解除されるので、プロセス間通信がより効率的になる。

【０００８】次に、第２のプロシージャは、テンポラリ・バッファ中の引数を処理することができる。引数を処理し終わったならば、第２のプロシージャは、リターン引数を整列させるために、第１メのモリスペース中で第２のバッファを割当てる。第２のプロシージャは、第１のプロシージャにリターンし、これに第２のバッファに対する第２の参照を渡す。第１のプロシージャは、この第１のプロシージャへのリターンを検出し、第２のバッファ中の引数を整列解除する。第１のプロシージャは、第２のバッファを割当て解除し、実行を継続する。本発明の実施の形態においては、第１のメモリスペースが第１の数のバッファを参照する第１のプロシージャによる第２のプロシージャの呼出しに先立って、第１のメモリスペースが第１のサイズに事前割当てされる。上記第１の数の各バッファを第１または第２のプロシージャによって割当て、それらのバッファが第１または第２のプロシージャによって使用中であることを指示することも可能である。上記第１の数の各バッファが第１のプロシージャまたは第２のプロシージャによって使用中であることの指示は、割当てフラグとバッファが使用中であることを示す値との間のアトミック・スワップによって行われ、その際上記割当てフラグは上記第１の数のバッファの１つが使用中であるかどうかを示すために用いられる。

【０００９】以下、本発明を添付図面に示す実施の形態により詳細に説明する。これらの実施の形態は、例示説明を目的とするものであり、本発明に対して制限的な意味を有するものではない。なお、添付図面中、同じ参照符号は同じ構成要素を示す。

【００１０】

【実施の形態】本発明は、プロセス間通信、特にリモート・プロシージャ呼出しを実装したコンピュータシステムにおけるプロセス間通信のより効率的な方法を提供するものである。本発明は、特定の実施の形態との関連、特に汎用プログラム・コンピュータシステムとの関連において説明するが、当業者には、本発明がその要旨及び範囲を逸脱することなく様々なシステムで実施すること可能なことは明白であろう。本発明は、一連のデータ構造及びコンピュータシステム中で動作可能なコンピュータ・プログラムに実装される付随命令として実施される。このようなデータ構造は、図２のブロック図に示すようなコンピュータシステムで生成することができる。

【００１１】図２において、符号２００は、本発明の一実施の形態が実装されるコンピュータシステムを示す。コンピュータシステム２００は、情報を伝達するためのバスまたはその他の通信手段２０１、バス２０１と結合された情報を処理するための処理手段２０２を具備する。システム２００は、さらに、情報及びプロセッサ２０２によって実行される命令を記憶するためのバス２０１に結合されたランダムアクセスメモリ（ＲＡＭ）またはその他の書換可能な記憶装置２０４（以下主メモリと称する）を具備する。また、主メモリ２０４は、プロセッサ２０２による命令実行の間、一時的変数またはその他の中間情報を記憶するためにも使用することができる。さらに、システム２００は、プロセッサ２０２用の静的情報及び命令を記憶するためのバス２０１と結合されたリードオンリーメモリ（ＲＯＭ）及び／またはその他の静的記憶装置２０６、及び磁気ディスクまたは光ディスクとこれらに対応するディスクドライブのようなデータ記憶装置２０７も具備している。データ記憶装置２

０７は、情報及び命令を記憶するために、バス２０１と結合されている。この記憶装置は、商業ベースで入手可能なソフトウェア製品を用いて、その時現在定義済みの問題記述についての情報を維持する以下に説明するデータベースの記憶に使用することも可能である。

【００１２】コンピュータシステム２００は、情報をコンピュータのユーザに表示するために、バス２０１に接続されたブラウン管（ＣＲＴ）や液晶表示装置（ＬＣＤ）のような表示装置２２１を具備することもできる。さらに、このような表示装置２２１は、表示装置２２１上に表示される単一または複数のフレームまたは画像のような情報を記憶するフレームバッファ２１０を介してバス２０１に接続することもできる。バス２０１には、情報及びコマンド・セレクションをプロセッサ２０２に伝達するための文字・数字キー及びその他のキーを含む文字・数字入力装置２２２を接続することも可能である。他のユーザ入力装置として、バス２０１には、方向情報及びコマンド・セレクションをプロセッサ２０２に伝達するとともに、表示装置２２１上のカーソルの動きを制御するためのマウス、トラックボール、スタイラス、あるいはカーソル方向キーのようなカーソル・コントロール２２３が接続されている。

【００１３】ここで、コンピュータシステム２００のコンポーネント及び関連ハードウェアは、その一部または全部を様々な態様で使用することができるが、各特定の態様により、種々の目的に応じて、システムのどのような構成でも使用することが可能であるということは理解できよう。

【００１４】一実施の形態においては、システム２００は、米国カリフォルニア州マウンテンビューのＳｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ（登録商標）社の製造になるＳＰＡＲＣｓｔａｔｉｏｎブランドのワークステーションのようなＳｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ（登録商標）のブランド・ファミリーのワークステーションの１つである。プロセッサ２０２としては、上記Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ（登録商標）社の製造になるＳＰＡＲＣブランドのマイクロプロセッサの一種を用いることができる。

【００１５】以下の様々な実施の形態の説明においては、高水準プログラミング言語（例えば、ＣまたはＣ＋＋言語）で生成され、例えば、上記の米国カリフォルニア州マウンテンビューのサン・マイクロシステムズ社より入手可能なＳＰＡＲＣｏｍｐｉｌｅｒによって実行時にシステム２００でコンパイル、リンクされた後、目的コードとして実行される一連のルーチンに具体的に言及する。詳しく言うと、本発明は、米国カリフォルニア州マウンテンビューのＳｕｎＳｏｆｔ社から入手可能なＳｏｌａｒｉｓ（登録商標）のスレッド・パッケージのような一部のソフトウェアライブラリと共に使用することができる（Ｓｕｎ、Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ及びＳｏｌａｒｉｓは、カリフォルニア州マウンテンビューの　Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ社の登録商標である。ＳＰＡＲＣ及びＳＰＡＲＣｓｔａｔｉｏｎは、スパーク・インターナショナル（ＳＰＡＲＣ　Ｉｎｔｅｒｎａｔｉｏｎａｌ，Ｉｎｃ．）社の登録商標であり、サン・マイクロシステムズ（Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ）社に専用実施権が許諾されている。しかしながら、以下に説明する方法及び装置は、離散型の論理デバイス、大規模集積回路（ＬＳＩ）、特定用途向け集積回路（ＡＳＩＣ）、またはその他の専用ハードウェアのような特殊目的のハードウェア装置に実装することも可能であるということは理解できよう。本願の説明は、本願と同様の機能を有する装置にも等しく適用可能である。

【００１６】次に、本発明の実施の形態について図３以降の図面を参照しつつ説明する。本発明の一実施の形態においては、プロセス間通信は２つのプロセス間の共用バッファスペースによって行われる。これらのプロセスは、１つの例においては、クライアント・プロセスとカーネルであり、他の場合においては、カーネルとサーバ・プロセスである。このようにして、クライアントとサーバ・プロセスの間の呼出しの時の引数の渡し操作のような通信を、以下に説明するメカニズムを用いて行うことができる。クライアント・プロセス、カーネル及びサーバを有するシステムにおいては、クライアントとサーバの間の通信を行うために２つのバッファ領域が使用される。これについては、図３及び４により図式的に説明する。図３に示すように、クライアント・プロセス３１０は、クライアント・プロセス３１０とカーネル・プロセス（ルーチン）３００の間の通信に使用する複数のバッファ（各々約５キロバイト）を割当てる。同様に、カーネル・プロセス３００とサーバ・プロセス３２０は、同様のサイズの複数のバッファ３２１を介して通信する。クライアント／カーネル間及びカーネル／サーバ間の通信には、「必要に応じて」、さらにバッファが割当てられる。このように、第１のプロセスでバッファを割当て、そのバッファの割当て状態を呼出しの間（通常呼出しからのリターンまで）維持する従来技術と異り、本発明の実施の形態においては、受け取り側のプロセスに呼出し側のプロセスから情報を受け取り次第、バッファを割当て解除させる。この動作を図４に図式化して示す。

【００１７】例えば、図４に示すように、第１のプロセス（例えば、図３の３１０）は、外部プロセスの呼出しと同時に引数を整列させ、バッファを割当てた後、その外部プロセスの呼出しを実行する。これは、図４のステップ４０１に示されている。場合によっては、外部プロセスは、「リモート」と呼ばれる。すなわち、この場合、外部プロセスは、第１のプロセス（クライアント）のアドレス空間へのアクセスを持たず、逆に第１のプロ

セスも外部プロセスのアドレス空間へのアクセスがない。同様に、分散型環境においては、このリモート・プロセスは、ローカル・プロセッサ、またはコンピュータシステムのメモリ、あるいはリモートのコンピュータシステムのリモート・プロセッサに常駐するものであってもよい。図３に示すカーネル・プロセス３００（別名「核」）による呼出しの検出と同時に、クライアントにより渡される引数を受け取るためのテンポラリ・バッファ（例えば図３の３０１）が割当てられる。クライアント・バッファからカーネル内のテンポラリ・バッファに引数がコピーされると同時に、クライアント・バッファは割当て解除される。このように、本実施の形態においては呼出し期間の間ずっとクライアント・バッファの割当て状態を維持するのではなく、クライアント・バッファは、カーネルが適切な引数を受け取り、コピーし終えるまでしか使用されない。

【００１８】カーネルに適切な引数がコピーされたならば、カーネルは引数を渡す第２のプロセスのためのサーバ・スレッドを得、カーネルとサーバの間の通信用に適切なバッファ（例えば図３の３２１）が割当てられる。引数がテンポラリ・バッファからサーバ・バッファへのコピーされた後、制御はサーバ・プロセス３２０に渡される。このステップは、図４の４０２に示す。サーバ・プロセス３２０による呼出しの検出と同時に、サーバは、サーバ・バッファ３２１から引数を受け取って整列解除する。この時、サーバは、そのサーバ／カーネル・バッファ３２１を割当て解除し、図４のステップ４０３に示すように実行することができる。

【００１９】サーバ・プロセスの実行終了と同時に、サーバ・バッファは割当てられ、クライアント・プロシージャへリターンするための引数を整列させることができる。次に、リターン引数は、サーバ／カーネル・バッファ３２１に整列させられ、サーバ・プロセス３２０からのリターンが行われる。カーネル３００によるリターンの検出と同時に、カーネル内でテンポラリ・バッファが再度割当てられ、そこにリターン引数がコピーされ、その後、サーバ・バッファ３２１は割当て解除される。すると、カーネルは、サーバ・スレッドを解放し、カーネル３００とクライアント３１０の間の通信のためにクライアント・バッファを割当てることができる。この点で、カーネルは、ステップ４０４で引数を逆にクライアント／カーネル・バッファ３１１にコピーし、制御のクライアント３１０へのリターンが行われる。ステップ４０５におけるカーネルからクライアントへの制御のリターンが検出されると、クライアントは、バッファ中にある引数を整列解除し、バッファが割当は解除される。次に、クライアントは、サーバ・プロセス３２０からリターン引数がリターンされた後、実行を継続することができる。バッファ３２１は、再び次のスレッドを実行可能な状態になる。

【００２０】このように、本発明においては、クライアントとサーバの間の通信を行うためのバッファが、「必要に応じて」動的に割当てられる。バッファは、２つのプロセス間の通信にとって必要でなくなると、「割当て済みバッファ」フラグのクリアによって割当て解除される。これは、第２のプロセス（例えばサーバ）からのリターンを待ってバッファを割当て解除する従来技術と対照的な特徴である。現代のコンピュータシステムに共通に用いられているようなマルチスレッド環境においては、各呼出しスレッド毎の別個のバッファの割当てのため、及び呼出しの間これらのバッファを割当て状態に維持するために多量のメモリ資源が費消される。通常、このような従来の技術的情況においては、サーバ・プロセスへの呼出し期間の間、これらのバッファは使用されない状態に置かれる。このように、本発明は、第２のプロセス（例えば、サーバ・プロセス３２０）の実行の間に必要とされない多数のスレッドにバッファを割当てることによる大量のメモリ使用を回避することによって、プロセス間通信のためのより効率的な手段を提供するものである。従って、本発明の実施の形態においては、そのような従来技術におけるプロセス間通信よりはるかに効率的にメモリが使用される。

【００２１】本発明の実施の形態では、このプロセス間通信を容易にするための図５に示すようなデータ構造を使用する。図５において、符号５００は制御域を示す。これは、例えば、クライアント／カーネル制御域（あるいはカーネル／サーバまたは他のその他の通信領域）であってもよい。この領域は、２つのプロセスの間に必要な通信を行うために用いられる。制御域５００は、どちらか通信を出す側のプロセスよってアクセスされるポインタを介して、複数の事前割当てされたバッファ５１３～５１９を参照する。これらは、プロセス間通信時にクライアント、カーネルまたはサーバ・プロセスによって実際にデータが格納されるバッファ領域である。制御域５００は、現在割当てられているバッファ数を表す整数値を入れる第１のフィールド５０１を有する。図５に示す例においては、フィールド５０１は整数値４が書き込まれており、現在４つのバッファ領域（５１３、５１５、５１７及び５１９）が割当てられていることを示している。本発明の実施の形態においては、フィールド５０１は、最大値として３２を書き込む（最大３２のバッファを参照する）ことができるが、これは単に設計上の選択に関する問題であり、使用するバッファの数は３２より多くても少なくてもよい。

【００２２】また、制御域５００は、所与のバッファ領域が現在割当てられているかどうかを示す割当てフラグを有する。図に示すように、フィールド５０２、５０４、５０６、５０８は、次のフィールド（ポインタまたは参照）が現在あるスレッドに割当てられているバッファまたはメモリエリアを指示しているかどうかを示す

「ａｌｌｏｃ／ｄｅａｌｌｏｃ」フラグを有する。フィールド５０３、５０５、５０７、５０９等は、バッファ自身へのポインタまたは参照であり、各々前のフィールドのａｌｌｏｃ／ｄｅａｌｌｏｃフラグと対応する。例えば、フィールド５０２中のフラグの値は、フィールド５０３中のポインタによって参照されたバッファ５１３が現在割当てられているか、割当て解除されているかを指示する。バッファ５１３～５１９は、各々小さいメモリエリアからなり、例えば、本発明の実施の形態においては、５キロバイトである。これらのバッファ領域は、フィールド５０１を調べて制御域によって参照された中で利用可能なバッファがあるかどうかを検知し、かつ各ａｌｌｏｃ／ｄｅａｌｌｏｃフラグ（例えば、５０２、５０４等）を調べて、その調べている特定のバッファが使用可能であるかどうかを決定することにより、クライアント、カーネルまたはサーバ・プロセスによって「必要に応じて」割当てられる。

【００２３】クライアント・プロセス（例えば図３の３１０）の初期化と同時に、第１のプロセス（例えばクライアント）から第２のプロセス（例えばカーネル３００）への何らかの通信時における使用のために制御域５００が割当てられる。現在使用中のバッファ数が生成中の全スレッドにとって不十分なときは、各々のバッファに使用される実メモリスペースも、「必要に応じて」オペレーティング・システムから割当てられる。図８に示すようなもう一つの実施の形態においては、ｎの全てバッファ（ｎ＝３２）に対してメモリを一度に割当てることが可能である（例えばクライアント・プロセスに入ると同時に、またはスレッドの最初の生成と同時に）。

【００２４】第２のプロシージャの呼出しが検出されると、以下に図６により説明するようにしてバッファを割当てることができる。この処理は、クライアント、カーネルまたはサーバが通信しているプロセスへ、あるいはそのプロセスから引数を渡すためにバッファを割当てるとき行うことができる。本発明の一実施の形態においては、バッファは、第２のプロセスの呼出しの間に常に使用可能である。本発明のもう一つの実施の形態においては、クライアントは、共用バッファのいずれかを使用する前に、最小量のメモリ（例えば１２８バイト）以上のメモリ量が必要かどうかを決定することも可能である。

【００２５】上記のいずれの場合にも、プロセス６００は、ステップ６０２で開始され、まずカウンタを図５の要素５０１のようなアレイＡ［０］中の第１の要素に等しくセットする。インデックスは１に初期化される。次に、ステップ６０４で、カウンタが０から許容最大バッファ数である３２までの指定範囲外であるかどうかが決定される。この範囲外であれば、ステップ６０６で、割当てプロセスからエラーがリターンされる。そうでなければ、ステップ６０８で、カウンタが正確に０に等しいかどうかが判断される。カウンタの内容０は、生成しようとするスレッドに現在利用可能な使用されていないバッファがないということを示し、プロセスはステップ６１０にリターンして、今回割当てに利用可能な空のバッファはないということを指示する。この場合、プロセスは、異常終了するか、バッファが利用可能になるまで待つか、あるいはバッファを含む第２のメモリエリアを割当てることができる。

【００２６】さらにプロセス６００の説明を続けると、ステップ６０４でカウンタの内容が上記範囲外でないと判断されるか、ステップ６０８で正確に０に等しくないことが検出された場合は、ステップ６１２で仮の値が０に等しくセットされる。上に説明した実施の形態で使用する規定においては、フィールド５０２、５０４等の１つの中の整数ゼロ（０）は、制御域５００の対応ポインタによって指示されるバッファが割当て済みであることを示す。そのフィールドの整数１は、バッファが割当てられていないということ指示する。ステップ６１２では、一時的変数ＴＥＭＰが０に等しくセットされる。次に、ステップ６１３で、Ａ［ｉｎｄｅｘ］中の割当てフラグがＴＥＭＰとアトミック・スワップされ、割当てフラグがクリアされる。本発明の実施の形態においては、ＳＰＡＲＣブランドのマイクロプロセッサ上で利用可能なような「アトミック・スワップ」動作を使用する。この動作は、原子的に、すなわち途中の割込み、据置きトラップ、またはシステム内の他のスレッドが割当てフラグＡ［ｉｎｄｅｘ］にアクセスするのを許容することなく行われる。このように、この領域にアクセスする他のプロセスは、値がスワップされ終わるまで、ロックアウトされる。アトミック・スワップについては、例えば、米国カリフォルニア州メンロパーク（Ｍｅｎｌｏ　Ｐａｒｋ）のＳＰＡＲＣ　Ｉｎｔｅｒｎａｔｉｏｎａｌ社から入手可能なＳＰＡＲＣアーキテクチャ・マニュアル（Ａｒｃｈｉｔｅｃｔｕｒｅ　Ｍａｎｕａｌ）（バージョン８、１９９２年刊）の１０２～１０３ページに記載されている。

【００２７】次に、ステップ６１４では、ａｌｌｏｃ／ｄｅａｌｌｏｃ（あるいはレジスタ）から取り出された値（ステップ６１４ではＴＥＭＰ変数）が１に等しいかどうかが判断される。この値が１に等しい場合は、バッファが割当てに利用可能であり、ステップ６１８で、バッファのポインタが要求プロセスにリターンされる。１に等しくない場合は、ステップ６１６で、カウンタが１だけディクリメントされ、次のａｌｌｏｃ／ｄｅａｌｌｏｃを調べるためにインデックスが２だけインクリメントされる。ステップ６０８～６１６は、制御域によって参照された各バッファについてａｌｌｏｃ／ｄｅａｌｌｏｃフラグをチェックすることによって、利用可能なバッファが検出されるまで続けられる。このように、２つのプロセス（例えば、クライアント／カーネルまたはカーネル／サーバ）間の通信のために指定された通信領域

からのバッファの割当てを容易に行うことができる。このプロセスは、図４によって上に説明した割当てステップにとって特に役に立つ。

【００２８】次に、バッファの割当て解除について図７のプロセス７００を参照して説明する。割当て解除プロセスは、ステップ７０２～７０８では前述のプロセス６００と同様に進行し、まず最初にバッファのポインタｂを受け取って、割当て済みバッファの数Ａ［０］を取り出し、ステップ７０２で、インデックスを１に初期化する。次に、ステップ７０４で、カウンタの内容が範囲内にあるかどうかがチェックされ、範囲内になければ、ステップ７０６で、エラーを伴うプロセスからのリターンが発生する。ステップ７０８では、制御域によって参照された中で割当て済みでないバッファがあるかどうかが判断される。割当て済みでないバッファがなければ、すなわちカウンタの内容が０に等しければ、ステップ７１０でエラーがリターンされる。ステップ７１２では、制御構造における参照により指示されたバッファが、割当て解除しようとするバッファのポインタｂに等しいかどうかが判断される。これが等しければ、対応する割当てフラグＡ［ｉｎｄｅｘ］が１に等しくセットされ、バッファが現在は割当て解除されていて、他のプロセスのために試用可能であるということを指示する。次に、プロセスは、ステップ７１８で、動作が首尾よく終了したことを示すリターン引数（例えばＯＫ）と共にリターンする。上記の対応するポインタが、ステップ７１２での判断に基づき、割当て解除しようとするバッファを指示しない場合は、ステップ７１４において、インデックスが２だけインクリメントされ、カウンタが１だけディクリメントされる。ステップ７０８～７１４は、割当て解除しようとするバッファがステップ７１２で決定されるか、ステップ７０８でチェックしようとするバッファがもはや残ったいなく（カウンタ＝０）なるまで繰り返される。

【００２９】次に、最後のプロセスとして、クライアントとカーネルの間、あるいはカーネルとサーバ・プロセスの間の通信のための共用メモリエリアの初期化のような、本発明の一実施の形態における２つのプロセス間の共用メモリエリア（図８のプロセス・フローチャート８００ではメモリエリアＡと称する）の初期化について説明する。この場合も、前述の場合同様に、ｎのバッファ全部の割当てをクライアント・プロセスに入ると同時に行うこともできれば、各バッファに必要に応じて個別にメモリを割当てることも可能である。図８に示すように、ステップ８０２においては、制御域５００の第１の要素（例えば、５０１）における利用可能なバッファの数Ａ［０］がｎに等しくセットされる。実施の形態においてはｎ＝３２であるが、設計上の選択事項に従って任意の数のバッファを使用することが可能である。次に、対応するカウンタもｎに等しくセットされ、インデックス変数が１に等しくセットされる。次に、ステップ８０４で、ｎの全てのバッファについてメモリがオペレーティング・システムから割当てられたかどうかどうかが判断される。その結果がノーの場合、プロセス８００はステップ８０８に進む。ステップ８０８では、対応するバッファのための割当てフラグが１に等しくセットされ、そのバッファが使用可能であることを指示する。さらに、バッファＡの対応参照［ｉｎｄｅｘ］＋１が、一部のオペレーティング・システムでａｌｌｏｃａｔｅ＿ｂｕｆｆｅｒ（）と命名されているようなメモリ割当て基本要素に等しくセットされる。この例においては、関数ａｌｌｏｃａｔｅ＿ｂｕｆｆｅｒは、例えば５キロバイト長のメモリ領域を割当てることができるが、バッファのサイズは、設計上の選択事項により任意の大きさとすることが可能である。

【００３０】ステップ８０８においてバッファ使用可能であるという指示が出、そのバッファに適切なメモリが割当てられたならば、インデックスが２だけインクリメントされ、カウンタは１だけディクリメントされる。プロセス・ステップ８０４～８１０は、クライアントとカーネルまたはカーネルとサーバのような２つのプロセス間の共用メモリ領域で総数ｎのバッファが適切なスペースを割当てられたことが検出されるまで繰り返される（割当てられるバッファの数がｎに等しい場合）。ステップ８０４でカウンタの内容が０に等しいことが検出された場合は、プロセスは終了し、ステップ８０６にリターンする。

【００３１】このように、本発明による上記の技術を用いることによって、２つのプロセスの間の通信のための制御域とバッファを生成し、プロセス間通信に用いることができる。本発明は、所与の任意の時点でいくつかのプロセス・スレッドがアクティブになり得るような情況において特に有用であり、メモリの使用を従来技術におけるよりも数段効率的にすることができる。以上、本発明に関する特定の実施の形態を特に図面を参照しつつ説明したが、当業者には、本発明の要旨及び範囲から逸脱することなく本発明の変更態様や修正態様を達成することが可能なことは明白であろう。従って、本発明は特許請求の範囲の記載によってのみ限定されるものである。

【図面の簡単な説明】

【図１】　従来技術によるプロセス間通信の方法を示すブロック図である。

【図２】　本発明の実施の形態を実装することができるコンピュータシステムを示すブロック図である。

【図３】　コンピュータシステムにおけるプロセス及びそれらの各プロセスに割当てられるバッファを示すブロック図である。

【図４】　本発明の実施の形態におけるクライアント・プロセス、カーネル及びサーバ・プロセス中の一連のステップを示す説明図である。

17

【図５】　プロセス間通信に使用されるバッファの詳細な構造を示す説明図である。

【図６】　図６は、２つのプロセス間で通信するために用いられる指定されたエリアの中でバッファを割当てるための方法を示すフローチャートである。

【図７】　２つのプロセス間で通信するための指定されたエリアの中でバッファを割当て解除するための方法を示すフローチャートである。

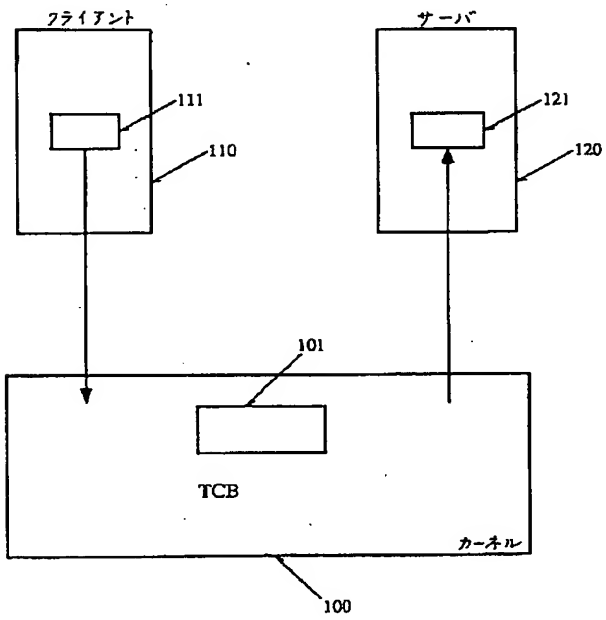
18

【図８】　メモリをプロセス間通信に割当てるための方法を示すフローチャートである。

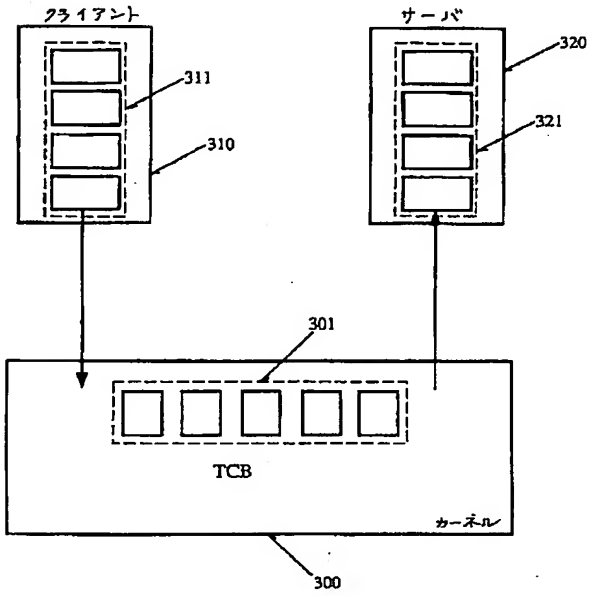
【符号の説明】

２００　コンピュータシステム、２０１　バス、２０４　主メモリ、２０６　ＲＯＭ、２０７　大容量記憶装置、２１０　フレームバッファ、２２１　表示装置、２２２　キーボード、２２３　カーソル・コントロール、２２４　ハードコピー装置。

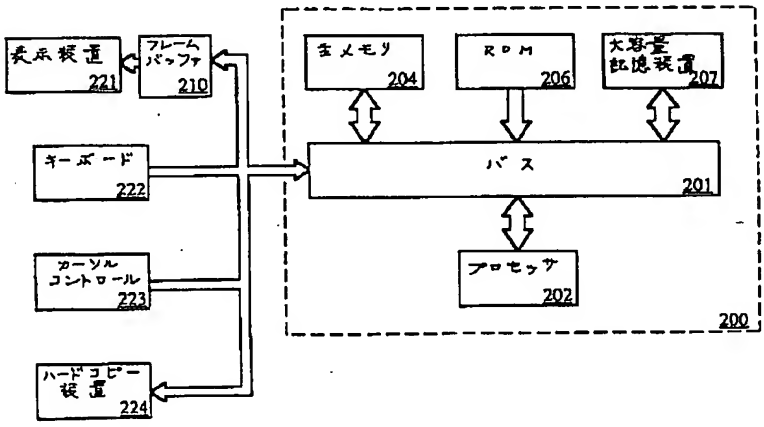
【図１】



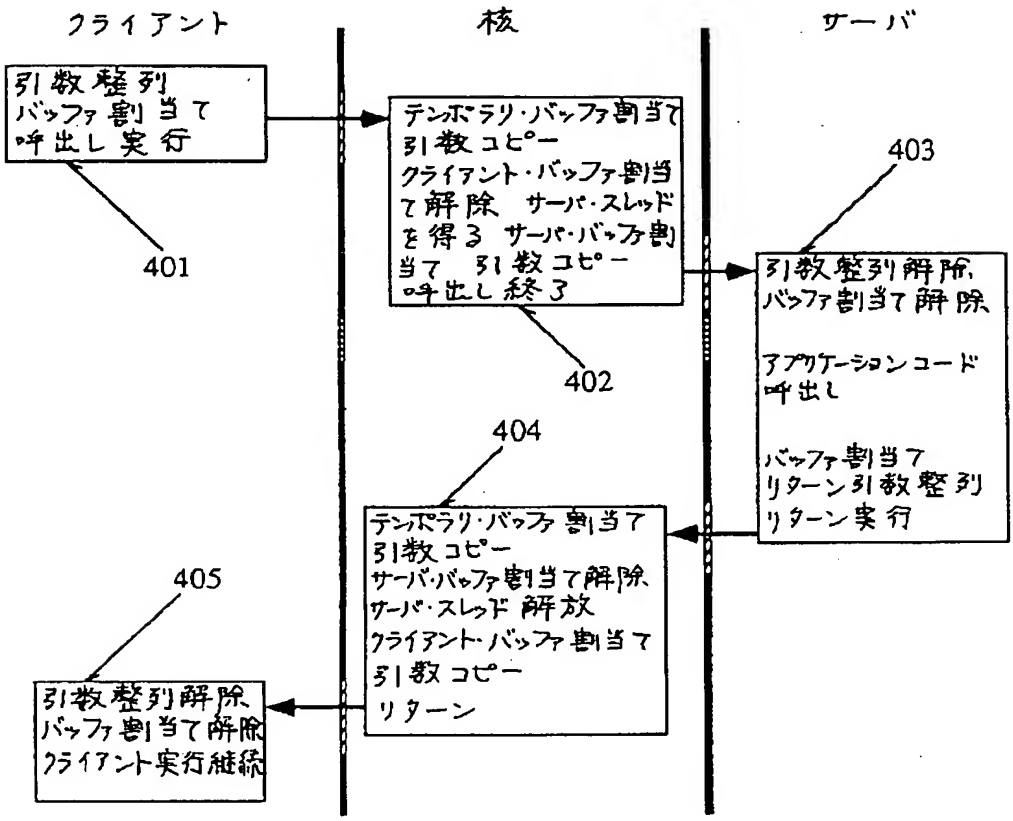
【図３】



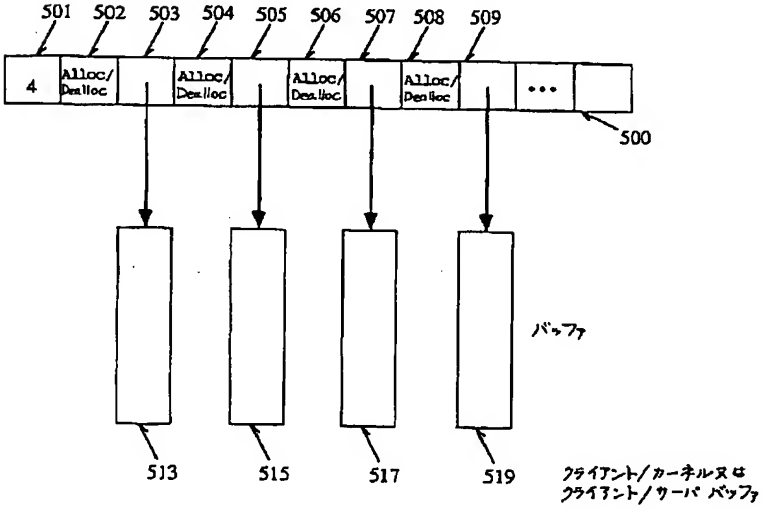
【図２】



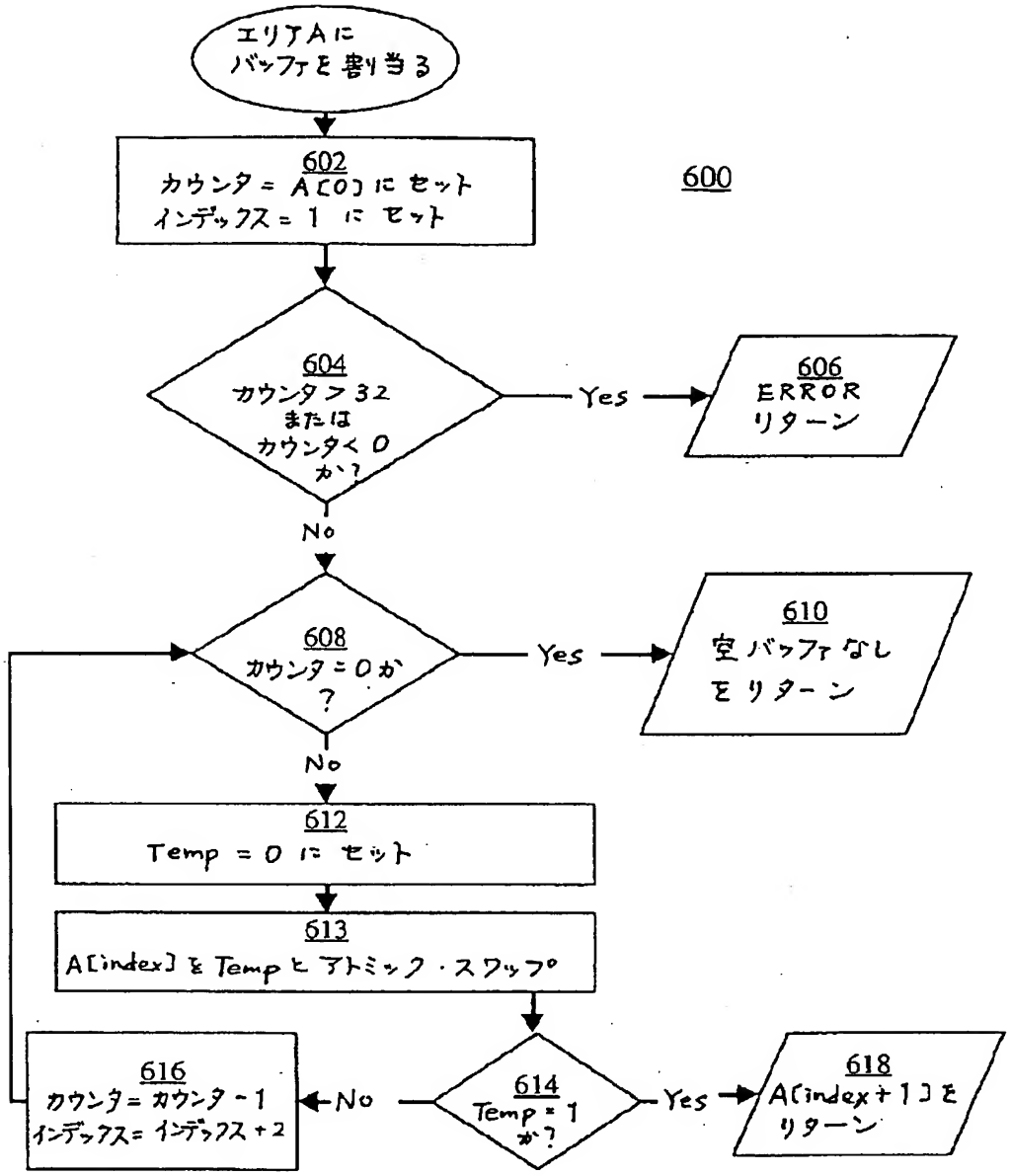
【図４】


クライアント
核
サーバ
引数整列
バッファ割当て
呼出し実行
401
テンポラリ・バッファ割当て
引数コピー
クライアント・バッファ割当て解除 サーバ・スレッドを得る サーバ・バッファ割当て 引数コピー
呼出し終了
402
403
引数整列解除、
バッファ割当て解除、
アプケーションコード
呼出し
バッファ割当て
リターン引数整列
リターン実行
404
テンポラリ・バッファ割当て
引数コピー
サーバ・バッファ割当て解除、
サーバ・スレッド解放
クライアント・バッファ割当て
引数コピー
リターン
405
引数整列解除、
バッファ割当て解除、
クライアント実行継続

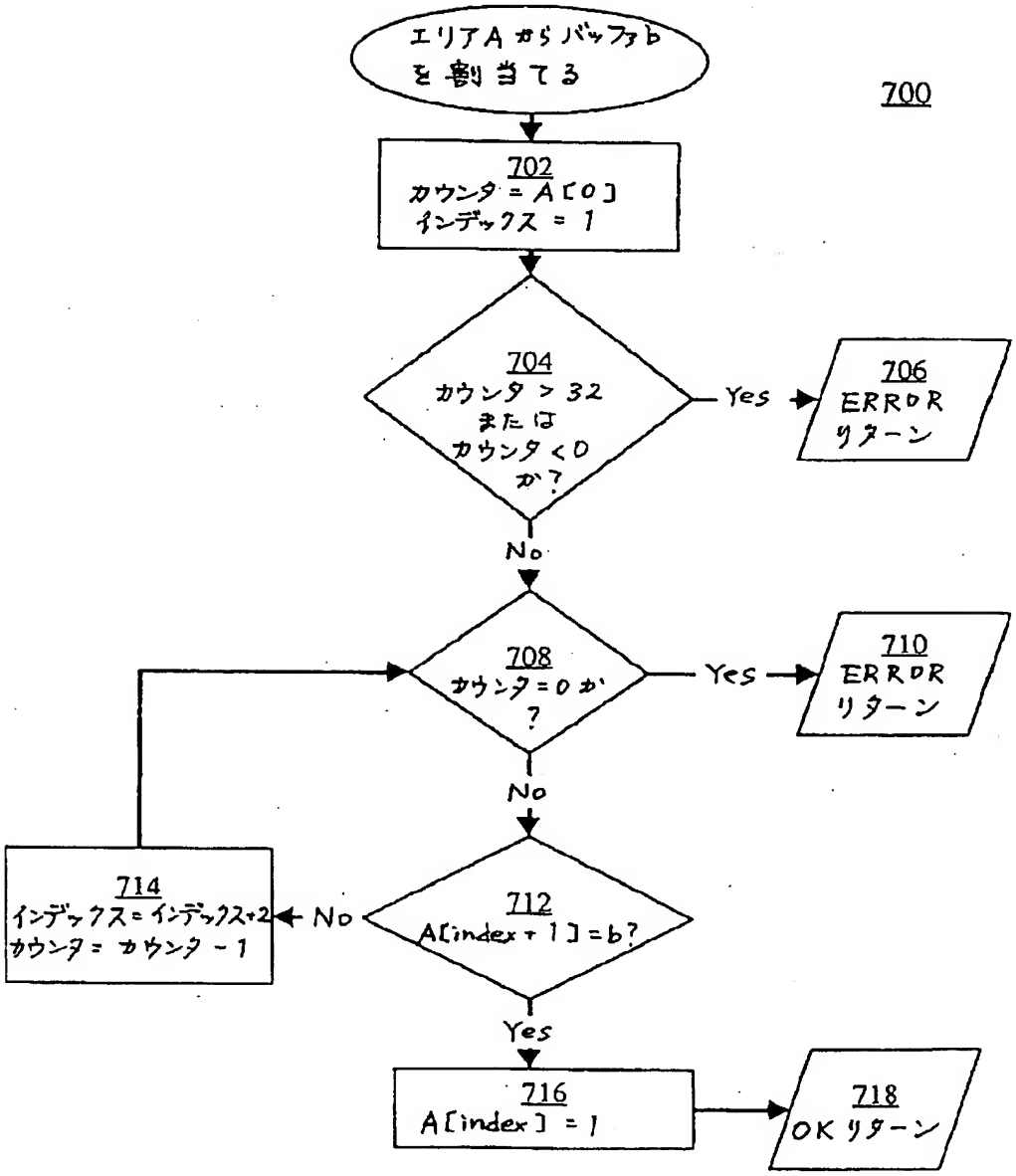
【図５】


501
502
503
504
505
506
507
508
509
4
Alloc/
Dealloc
Alloc/
Dealloc
Alloc/
Dealloc
Alloc/
Dealloc
...
500
バッファ
513
515
517
519
クライアント/カーネルまたは
クライアント/サーバ バッファ

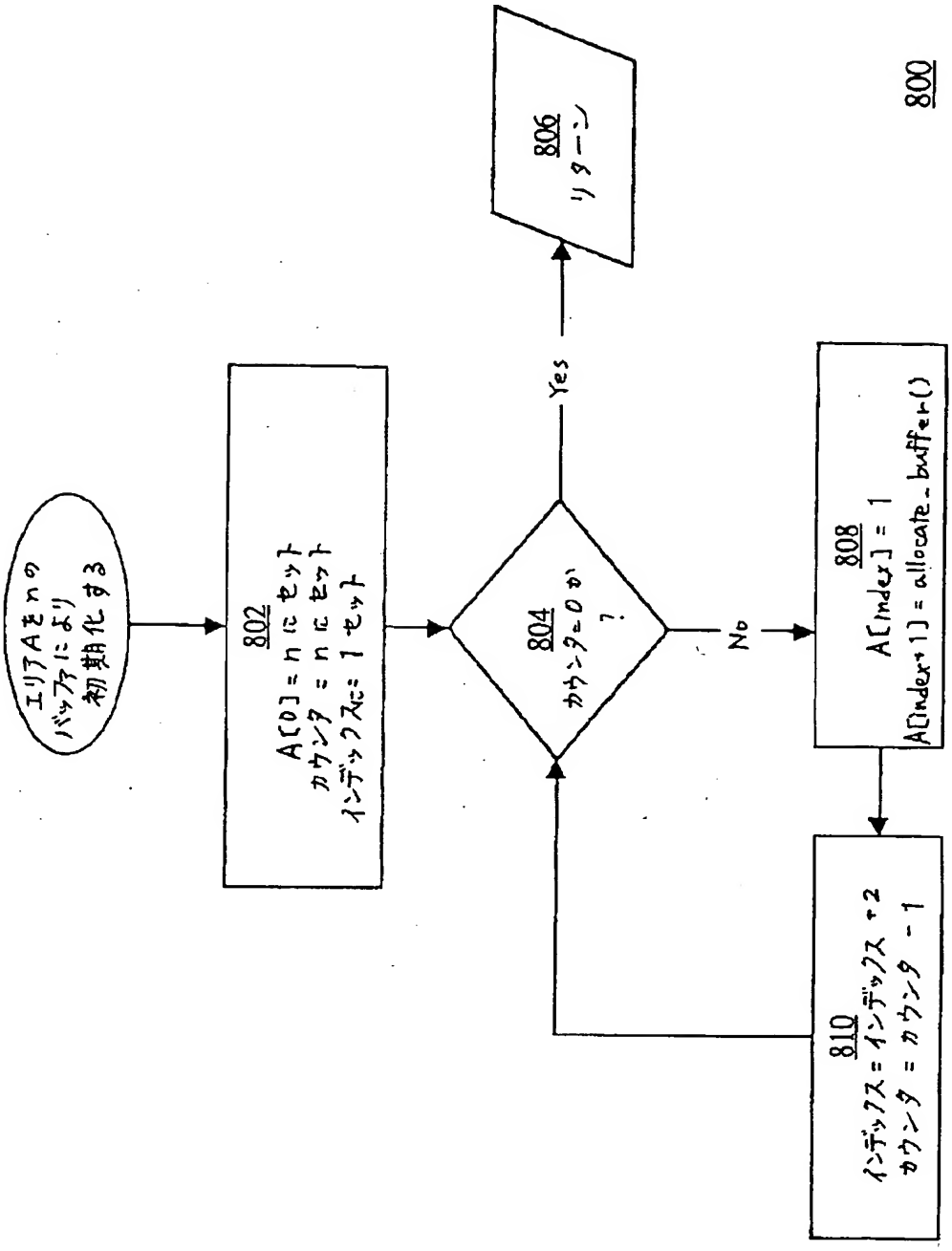
【図６】


エリアAに
バッファを割当る
602
カウンタ = A[0] にセット
インデックス = 1 にセット
600
604
カウンタ > 32
または
カウンタ < 0
か？
Yes
606
ERROR
リターン
No
608
カウンタ = 0か
？
Yes
610
空バッファなし
をリターン
No
612
Temp = 0 にセット
613
A[index] と Temp とアトミック・スワップ
614
Temp = 1
か？
No
616
カウンタ = カウンタ - 1
インデックス = インデックス + 2
Yes
618
A[index + 1] を
リターン

【図７】


エリアAからバッファbを割当てる
700
702
カウンタ＝A[0]
インデックス＝1
704
カウンタ＞32
または
カウンタ＜0
か？
Yes
706
ERROR
リターン
No
708
カウンタ＝0か？
Yes
710
ERROR
リターン
No
712
A[index＋1]＝b？
No
714
インデックス＝インデックス＋2
カウンタ＝カウンタ－1
Yes
716
A[index]＝1
718
OKリターン

【図８】



フロントページの続き

(72)発明者　グラハム・ハミルトン
アメリカ合衆国　94303　カリフォルニア
州・パロ　アルト・デビッド　コート・
3143